Resource-constrained project scheduling for timely project completion with stochastic activity durations

Francisco Ballestín
Department of Statistics and Operations Research
Universidad Pública de Navarra, Spain
Tel. +34 948 169 216; Fax +34 948 169 204
Francisco.Ballestin@unavarra.es

Roel Leus
Department of Decision Sciences and Information Management
Katholieke Universiteit Leuven, Belgium
Tel. +32 16 32 69 67; Fax +32 16 32 67 32
Roel.Leus@econ.kuleuven.be

Resource-constrained project scheduling for timely project completion with stochastic activity durations

We investigate resource-constrained project scheduling with stochastic activity durations. Various objective functions related to timely project completion are examined, as well as the correlation between these objectives. We develop a GRASP-heuristic to produce high-quality solutions, using so-called descriptive sampling. The algorithm outperforms existing algorithms for expected-makespan minimization. The distribution of the possible makespan realizations for a given scheduling policy is also studied.

Keywords: project scheduling; resource constraints; uncertainty; stochastic activity durations; GRASP.

History: Submitted to *Production and Operations Management* in July 2007; Revisions February 2008 and September 2008; Accepted September 2008.

---

1. Introduction

The larger part of the scientific literature on resource-constrained project scheduling focuses on the minimization of project duration in a deterministic setting. The goal of the resource-constrained project scheduling problem (RCPSP) is to minimize the duration of a project subject to finish-start precedence constraints and renewable resource constraints. It is shown in Blazewicz et al. (1983) that the RCPSP, as a job-shop generalization, is NP-hard in the strong sense. A large number of exact and heuristic procedures have been proposed to construct workable baseline schedules that solve this deterministic RCPSP; see Demeulemeester and Herroelen (2002), Kolisch and Padman (2001) and Neumann et al. (2002) for recent overviews and Herroelen (2005) for a discussion on the link between theory and practice.

During project execution, however, project activities are often subject to considerable uncertainty, which derives from many different possible sources: activities may take more or less time than originally estimated, resources may become unavailable, materials may arrive behind schedule, workers may be absent, etc. In this article, we examine the case where this uncertainty is important enough to be incorporated into the planning phase. The sources of variability in processing times are manifold; nevertheless, the main scheduling objectives are mostly functions of the activities' start (or end) times, the project makespan being the single most-studied objective, in addition to other ones such as weighted earliness-tardiness and net present value of the project. This justifies a restriction to the study of

uncertainty in processing times only, although this variability may be generated by many causes. The stochastic resource-constrained project scheduling problem (stochastic RCPSP or SRCPSP) is the stochastic equivalent of the RCPSP, where activity durations are not known in advance but are represented as random variables. The probability distributions can either be objective (a risk situation) or result from subjective judgment (in the case of decision-theoretic uncertainty or even ignorance).

The SRCPSP usually aims at minimizing the expected makespan over a limited set of possible decisions to be made during project execution. As coherently described by Stork (2001), an important new aspect comes into play when we move from the deterministic to the stochastic case: what is a solution to an SRCPSP-instance? A deterministic schedule does not necessarily contain enough information to make decisions during project execution. Hence, for each possible event occurring during project execution, a solution should define an appropriate action, typically the start of new activities. To make such decisions, one may want to exploit the information given by the current state of the project. In line with Igelmund and Radermacher (1983), among others, we call such a solution a (scheduling) *policy*.

A vast amount of literature exists on the so-called (generalized) PERT-problem, where no resource constraints are taken into consideration. These studies are usually concerned with the computation of certain characteristics of the project makespan (earliest project completion), mainly with exact computation, approximation and bounding of the distribution function and the expected value. Note that in this case, no real scheduling effort is required: all activities can be started when their predecessors are completed. For a review of research up until 1987, we refer to Adlakha and Kulkarni (1989). A recent computational study on bounding the makespan distribution, in which the most promising algorithms are compared, was conducted by Ludwig et al. (2001).

Research into SRCPSP, however, has remained limited to date, with few computational publications addressing this problem: Igelmund and Radermacher (1983) and Stork (2001) report on experiments with branch-and-bound algorithms, while Golenko-Ginzburg and Gonik (1997) and Tsai and Gemmill (1998) develop greedy and local-search heuristics. Time/resource trade-offs with stochastic activity durations, in which resource allocation influences the mean and/or the variance of the durations, are investigated in Gerchak (2000), Gutjahr et al. (2000) and Wollmer (1985).

The contributions of this article are fivefold: (1) we examine multiple possible objective

functions for project scheduling with stochastic activity durations; (2) using computational experiments, we show that these different objective functions are closely connected and that, for most practical purposes, it may suffice to focus on minimizing the expected makespan; (3) we develop a GRASP-heuristic that produces high-quality solutions, outperforming existing algorithms for expected-makespan minimization; (4) the variance-reduction technique of descriptive sampling is applied and its benefits assessed; and (5) the distribution of makespan realizations for a given scheduling policy is studied.

The remainder of this article is organized as follows. Definitions and a detailed problem statement are provided in Section 2, followed by a discussion of the computational setup (Section 3). Section 4 presents the basic ingredients of our GRASP-algorithm. Our main computational results for the expected-makespan objective can be found in Section 5; the relationship between the expected makespan and some other objective functions is treated in Section 6. The distribution of the makespan realizations is the subject of Section 7. Finally, a summary is given in Section 8.

2. Definitions and problem statement

This section contains a number of definitions (Section 2.1), a discussion of scheduling policies (Section 2.2), and a statement of the problems to be solved (Section 2.3).

2.1 Definitions

A project consists of a set of activities $N = \{0, 1, \ldots, n\}$, which are to be processed without interruption on a number K of renewable resource types with availability a_k, $k = 1, \ldots, K$; each activity i requires $r_{ik} \in \mathbb{N}$ units of resource type k. The duration D_i of activity i is a random variable (r.v.); the vector $(D_0, D_1, \ldots, D_n)$ is denoted by $\mathbf{D}$. The set A is a (strict) partial order on N, i.e. an irreflexive and transitive binary relation, which represents technological precedence constraints. (Dummy) activities 0 and n represent start and end of the project, respectively, and are the (unique) least and greatest element of the partially ordered set (N, A). Activities 0 and n have zero resource usage and $Pr[D_i = 0] = 1$ for $i = 0, n$; for the remaining activities $i \in N \backslash \{0, n\}$ we assume that $Pr[D_i < 0] = 0$ ($Pr[e]$ represents the probability of event e). We associate the directed acyclic graph $G(N, A)$ with the partially ordered set (N, A).

We use lowercase vector $\mathbf{d} = (d_0, d_1, \ldots, d_n)$ to represent one particular realization (or sample, or scenario) of $\mathbf{D}$. Alternatively, when each duration D_i is a constant, which is the case in the (deterministic) RCPSP, we use the same notation $\mathbf{d}$. A solution for the RCPSP is a schedule $\mathbf{s}$, i.e., a vector of starting times $(s_0, s_1, \ldots, s_n)$ with $s_i \geq 0$ for all $i \in N$, that is both time-feasible and resource-feasible. Schedule $\mathbf{s}$ is called *time-feasible* if $s_i + d_i \leq s_j$ for all $(i, j) \in A$; $\mathbf{s}$ is said to be *resource-feasible* if, at any time t and for each resource type k, it holds that $\sum_{i \in \mathcal{A}(\mathbf{s},t)} r_{ik} \leq a_k$, where the *active set* $\mathcal{A}(\mathbf{s}, t) = \{i \in N | s_i \leq t < s_i + d_i\}$ contains the activities in $N \backslash \{0, n\}$ that are in progress at time t. The objective function of the RCPSP is the project makespan s_n (which is to be minimized).

2.2 Scheduling policies

The execution of a project in the context of the SRCPSP can best be seen as a dynamic decision process. A solution, then, is a *policy* Π, which defines *actions* at *decision times*. Decision times are typically $t = 0$ (the start of the project) and the completion times of activities. An action can entail the start of a set of activities that is precedence and resource feasible. A schedule is thus gradually constructed through time. A decision at time t can only use information available before or at time t; this requirement is often referred to as the *non-anticipativity constraint*. As soon as all activities are completed, activity durations are known, yielding a realization $\mathbf{d}$ of $\mathbf{D}$. Consequently, every policy Π may alternatively be interpreted (cf. Igelmund and Radermacher, 1983; Stork, 2001) as a function $\mathbb{R}_{\geq}^{n+1} \mapsto \mathbb{R}_{\geq}^{n+1}$ that maps given samples $\mathbf{d}$ of activity durations to vectors $\mathbf{s}(\mathbf{d}; \Pi) \in \mathbb{R}^{n+1}$ of feasible activity starting times (schedules); if no misinterpretation is possible, we usually omit the identification of the policy and write $\mathbf{s}(\mathbf{d})$. For a given scenario $\mathbf{d}$ and policy Π, $s_n(\mathbf{d}; \Pi)$ denotes the makespan of the schedule. The most-studied objective for the SRCPSP is the selection of a policy Π^* within a specific class that minimizes $E[s_n(\mathbf{D}; \Pi)]$, with $E[\cdot]$ the expectation operator with respect to $\mathbf{D}$.

A well-known class of scheduling policies is the class of priority policies, which order all activities according to a priority list and, at every decision point t, start as many activities as possible in the order dictated by the list, which is in line with the parallel schedule generation scheme (parallel SGS) (see Kolisch and Hartmann, 1999). These list-scheduling policies have a number of drawbacks. First of all, priority policies cannot guarantee an optimal schedule. Moreover, changes in activity duration may lead to so-called Graham anomalies (Graham, 1966), such as increasing project duration due to decreasing activity duration. Stork (2001)

describes how, if we interpret a policy as a function, these anomalies lead us to conclude that priority policies are neither monotone nor continuous.

Several other classes of policies have been examined by Stork (2001), most of which exhibit severe computational limitations; he concludes that, for larger instances, the only remaining alternative is to use the class of so-called activity-based policies, which is also the class that will be studied in this paper (Stork uses the term 'job-based (priority) policies'). An activity-based policy $\Pi(L)$ is also represented by a priority list L of the activities and, for a given sample $\mathbf{d}$, computes starting times by starting each activity in the order imposed by L as early as possible, with the side constraint that $s_i(\mathbf{d}) \leq s_j(\mathbf{d})$ if $i \prec_L j$. Elimination of this side constraint would yield a simple priority policy that suffers from the Graham anomalies, but the 'activity-based' point of view, rather than the greedy 'resource-based' one, eliminates this problem. Since these activity-based policies perform activity incrementation rather than time incrementation, they are alternatively referred to as '(stochastic) serial SGS' (Ballestín, 2007).

We provide a small example to illustrate the difference between activity-based policies and simple priority policies. Consider a project with activity set $N = \{0, \ldots, 4\}$ containing three non-dummy activities 1, 2 and 3; for the precedence constraints we have $A(\{1, 2, 3\}) = \{(1, 2)\}$, with $A(E)$ the edges of the subgraph of $G(N, A)$ that is induced by $E \subseteq N$. There is one resource type ($K = 1$), with $r_{i1} = 1$ for $i = 1, 2, 3$, and availability $a_1 = 2$. The list L corresponding with $0 \prec_L 1 \prec_L 2 \prec_L 3 \prec_L 4$ can be used to define either a priority policy or an activity-based policy. The priority policy will start both activity 1 and 3 at time 0, while the activity-based policy will not start activity 3 earlier than the starting time of activity 2 because $2 \prec_L 3$.

2.3 Problem statement

The literature on project management abounds with motivations for reducing project lead times, including various first-mover advantages in new-product development (see, for instance, Smith and Reinertsen, 1991), advantages during the bidding process (Kerzner, 1998; Newbold, 1998; Xie et al., 2006), and incentive contracts that contain a penalty for delayed completion or a bonus for early delivery (Bayiz and Corbett, 2005). This justifies our focus on timely project completion by using some characterization of the makespan (which is a stochastic variable) as an objective function. We elaborate on this choice in the paragraphs below.

2.3.1 Individual projects

French (1988) describes four criteria for decision making under uncertainty. For a minimization problem, these amount to (1) *minimax* (minimize the worst possible makespan realization), (2) *minimin* (minimize the best possible outcome, which is an optimistic approach, as opposed to the pessimistic minimax), (3) *minimax regret* (minimize the largest possible difference in makespan between the policy to be selected and the optimal makespan for a given realization), and (4) minimize the objective *in expectation*. Scheduling with objectives (1) and (3) is studied in Kouvelis and Yu (1997); we do not adopt these objectives because (a) one normally needs discrete scenarios instead of continuous distributions, and (b) the optimization problem with constant durations should be easy (solvable in polynomial time) in order to produce computational results for average-size instances (in the case of Kouvelis and Yu: single-machine scheduling with total-flow-time objective and two-machine flow-shop with makespan objective). Since a practical decision maker is usually risk-averse, we also do not investigate objective (2).

In conclusion, when a project is to be executed in isolation (e.g. for internal clients), the expected makespan (subsequently also called the *expectation* objective) is the most logical objective of the foregoing. Actually, it is well known that the expectation criterion is most appropriate for a *risk-neutral* decision maker. In order to represent possible risk averseness, we opt for constraints of the form $Pr[s_n(\mathbf{D}) \geq \delta] \leq p$, for given probability p and deadline δ, rather than for utility functions with their inherent difficulty of estimation. This reflects the undesirability of exceptionally high makespan realizations and is comparable with downside-risk or Value-at-Risk (VaR) constraints in Finance (Ang et al., 2006; Jorion, 2000). Likewise, Schuyler (2001) also advocates conservatism when associating large potential gains and losses with individual decisions. Henceforth, in line with Portougal and Trietsch (1998), $\max Pr[s_n \leq \delta]$ is referred to as the *service-level* objective due to its similarity with inventory management (see, for instance, Silver et al., 1998). For the benefit of risk averseness, a lower bound may be imposed on the service level. In the remainder of this text, we use the term *statistic* to refer to any function of s_n. Obviously, a decision maker may be interested in possible trade-offs between different statistics; the correlation between statistics will be studied in Section 6.

A second way to account for risk averseness is to investigate the trade-off between the expected makespan $E[s_n(\mathbf{D})]$ and the makespan variance $\text{var}[s_n(\mathbf{D})]$. This is in line with

Portougal and Trietsch (1998), who suggest that "variance reduction should be introduced explicitly in the objective, while retaining the expected completion time as well". Similarly, Elmaghraby et al. (1999) also distinguish both mean and variance of the project duration as the two prime performance measures of concern. Gutierrez and Paul (2001) examine the impact of variability in activity duration on mean project duration, while Cho and Yum (1997) focus on the sensitivity of makespan variability. Finally, knowledge of the entire distribution function of makespan realizations for a given policy is obviously also highly informative to the decision maker; we investigate this in a separate section (Section 7).

2.3.2 External clients

The foregoing discussion looked into the execution of a project in isolation. When a project deadline has been negotiated with external clients beforehand, however, it may be more useful to adapt the scheduling objective function in order to reflect existing penalty structures (we do not focus on bonuses for early completion), which take the form either of a fixed charge, so that the objective function becomes $\max Pr[s_n \leq \delta]$ for a given deadline δ, or of a fixed charge *per unit-time overrun*, leading to $\min E[\max\{0; s_n - \delta\}]$ (the *tardiness* objective). We refer to Gutjahr et al. (2000) for a model that uses more general loss functions in a slightly different context.

3. Computational setup

3.1 General setting

The analyses in the next sections are based on computational experiments using randomly generated datasets. The coding was performed in C using the Microsoft Visual C++ 6.0 programming environment, and the experiments were run on a Samsung X15 Plus portable computer with Pentium M processor with 1,400 MHz clock speed and 512 MB RAM, equipped with Windows XP. Our tests were performed on instances from the benchmark library PSPLIB[1], which contains instances of different size and with different characteristics of the deterministic RCPSP, and which were generated by the problem generator ProGen (Kolisch and Sprecher, 1996); the number of resource types $K \leq 4$. We only use the dataset containing 600 instances with 120 activities (commonly named 'j120'). The deterministic duration d_i^* for each activity $i \neq 0, n$ is an integer randomly chosen from $\{1, 2, \ldots, 10\}$.

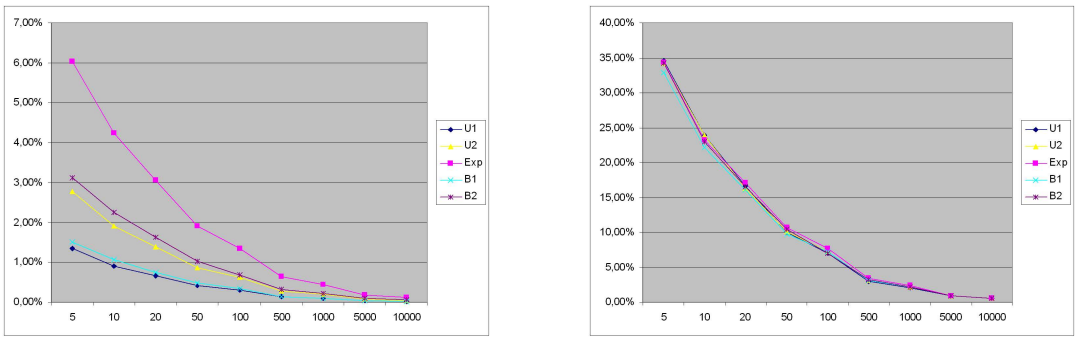
[1] Available at http://129.187.106.231/psplib/.

We generate the probability distributions of the stochastic activity processing times D_i (which are not created by the ProGen instance generator) in line with Stork (2001): we take the given deterministic processing time d_i^* of each activity as expectation and we construct Uniform, Exponential and Beta distributions. More specifically, we examine five distributions: a (continuous) Uniform distribution on $[d_i^* - \sqrt{d_i^*}; d_i^* + \sqrt{d_i^*}]$ (subsequently referred to as case 'U1'); (continuous) Uniform on $[0; 2d_i^*]$ ('U2'); Exponential with expectation d_i^* ('Exp'); Beta distribution with support $[d_i^*/2; 2d_i^*]$ and variance $= d_i^*/3$ ('B1'); and Beta with support $[d_i^*/2; 2d_i^*]$ and variance $= d_i^{*2}/3$ ('B2'). These five distributions have variances of $d_i^*/3$, $d_i^{*2}/3$, d_i^{*2}, $d_i^*/3$ and $d_i^{*2}/3$, respectively. The distributions have been created so that U1 and B1, on the one hand, and U2 and B2, on the other hand, share the same variance.

3.2 Evaluation of the statistics

Exact evaluation of the statistics under consideration (expectation, variance, ...) would be overly time-consuming (for the expected makespan, for instance, this amounts to the PERT-problem, which is well known to constitute a formidable computational challenge, see Hagstrom, 1988), which is why we approximate these values by means of simulation. Similar decisions in the context of scheduling under uncertainty have been made by Ballestín (2007), Leus and Herroelen (2004), Möhring and Radermacher (1989) and Stork (2001), among others. A solution in this article is an activity-based policy $\Pi(L)$, which is represented by an activity list L. An approximation of any statistic $g(L)$ associated with $\Pi(L)$ is based on sampling a number of realizations from $\mathbf{D}$; the number of replications is a parameter. Stork (2001), for instance, states that, for the expected makespan, 200 samples turn out to provide a reasonable trade-off between precision and computational effort.

For our computational experiments, we examined the standard deviation of the percentage deviation of simulated versus 'true' makespan, the latter obtained from a high number (25,000) of runs over the dataset. For a justification of the standard deviation as an accuracy measure and for a discussion of the convergence of the estimate towards the true value as the number of samples increases, see Kleywegt et al. (2001). Leus and Herroelen (2004) note that the number of simulation runs corresponding to the same standard deviation decreases with the number of activities; this approach has the advantage of reducing (relative) simulation effort for larger problem instances. For instances with 120 activities, our observations are summarized in Figure 1, which depicts the accuracy of our estimate of the expectation and standard deviation of the makespan (averaged over the 600 instances); the corresponding



(a) Expected makespan.

(b) Makespan standard deviation.

Figure 1: Accuracy, measured using the standard deviation of the percentage error, as a function of the number of replications.

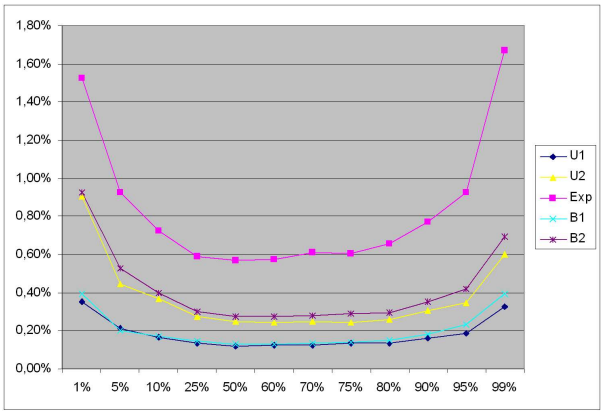


Figure 2: Accuracy of approximation of the service level for various due dates.

graph for the service level is very similar to Figure 1(a), the graph for the expected tardiness is close to Figure 1(b). Especially in Figure 1(a), the same number of replications clearly leads to less accurate results for distributions with higher variability. We conclude that 1,000 replications lead to an accuracy level of below 1% for the expected makespan. Note that the curves in Figures 1(a) and 1(b) are all more or less proportional to $1/\sqrt{z}$, if z represents the number of replications.

For 1,000 replications, Figure 2 shows the quality of the approximation of the due dates corresponding with different service levels (estimated based on the order statistics). The U-shaped graphs indicate that it is easier to approximate due dates corresponding with service levels in the middle of the interval $[0; 100\%]$ than in the tails; this will be elaborated in Section 6.

Ballestín (2007) shows that using fewer scenarios to calculate the approximation of each

activity list during the algorithm favors the calculation of more policies and leads to better solutions at the end of the procedure. Those scenarios were calculated using random sampling (simple Monte-Carlo sampling), as is common in most scenario-based algorithms. In this paper, we use *descriptive sampling* (Saliby, 1990, 1997), which is a variance-reduction technique. Descriptive sampling is based on a deterministic and purposive selection of sample values in order to match the sampled distribution as closely as possible, and on the random permutation of these values. More specifically, we generate quantiles for each activity duration and sample from that set of values. Concretely, 1,000 replications is the default number we work with for computing correlations and fitting distributions (Sections 6 and 7). On the other hand, when our focus is on the highest-quality solutions obtainable with a given computational effort, we consider the number of replications as a parameter of the algorithm. In our computational results in Section 5, we opt for 10 replications (which was the best tested number in Ballestín, 2007), and we provide empirical evidence showing that such a low number is preferable.

4. GRASP

Below, we discuss GRASP as a general heuristic procedure (Section 4.1) and we describe the overall structure of our search procedure for SRCPSP-solutions (Section 4.2).

4.1 GRASP as a general-purpose metaheuristic

A greedy randomized adaptive search procedure (GRASP) is a multi-start or iterative process (Aiex et al., 2002; Feo and Resende, 1995; Feo et al., 1994). Each GRASP-iteration consists of two phases: in a construction phase, a feasible solution is produced and, in a local-search phase, a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result.

In the construction phase, a feasible solution is iteratively constructed, one element at a time. The basic construction phase in GRASP is similar to the semi-greedy heuristic proposed independently by Hart and Shogan (1987). At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list $\mathcal{C}$ with respect to a greedy function $\mathcal{C} \mapsto \mathbb{R}$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at

Algorithm 1 GRASP: global algorithmic structure

```
1: EliteSet = ∅
2: while TerminationCriterion not met do
3:    L = BuildActList(EliteSet)
4:    s* = s(d*, Π(L))
5:    s* = LocalSearch(s*)
6:    L = ScheduleToList(s*)
7:    Evaluate the activity-based policy Π(L)
8:    if L is better than the worst solution L' in EliteSet then
9:       EliteSet = (EliteSet \L') ∪L
10:   end if
11: end while
12: Return the best solution found
```

each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP resides in the fact that we choose one of the best candidates in the list but not necessarily the top candidate; the list of best candidates is called the restricted candidate list. It is almost always beneficial to apply a local-search procedure to attempt to improve each constructed solution.

4.2 Adapting GRASP to our setting

The global structure of our GRASP-implementation is represented as Algorithm 1. Our basic algorithm maintains a set EliteSet of elite solutions (activity lists), containing the best solutions so far encountered. At each iteration of the algorithm, the solutions in EliteSet are used to create a new activity list with the procedure BuildActList. Subsequently, a schedule $\mathbf{s}^*$ is built by applying the serial SGS with the mean durations $\mathbf{d}^*$ to this list (the serial SGS schedules the activities as early as possible in the order of the list). After a local-search procedure that attempts to improve this deterministic schedule, we re-translate schedule $\mathbf{s}^*$ into an activity list by means of function ScheduleToList, ordering activities by starting times. This yields the representation of a new activity-based policy, whose objective function is then evaluated (using simulation). If it is better than the worst solution in the elite set, we erase that worst solution and include the new one.

There are two main differences between our algorithm and a 'standard' GRASP-implementation. First of all, the rule applied in standard GRASP to select the next element to be incorporated into the partial solution is the same throughout the algorithm. Our algorithm changes this rule dynamically, because it is based on the best solutions found, which usu-

Algorithm 2 BuildActList(EliteSet)

```
i = 0; EligibleSet = {0}; nit = 0
while i < n do
  if nit = 0 then
    reference = SelectSolution(EliteSet)
    if reference ≠ "LFT", "random" then
      nit ∈ [nitmin ; nitmax]
    end if
  else
    nit = nit −1
  end if
  Select an activity j from EligibleSet according to the reference
  L(i) = j; i = i + 1
end while
Return the activity list L
```

ally change during (at least part of) the search. For the RCPSP, it has been observed (see e.g. Hartmann and Kolisch, 2000) that self-learning procedures are better than procedures that do not learn from previously evaluated schedules. This is also true for the SRCPSP (Ballestín, 2007), where a genetic algorithm outperforms a sampling procedure.

The second difference with standard GRASP is that a set of solutions is maintained rather than just one. Our main motivation for this approach is that the best algorithms for RCPSP and for many other project-scheduling algorithms are population-based. Including several solutions thus favors the diversification of the search, as each solution in the set will tend to attract new solutions to its neighborhood. In our implementation, the cardinality of EliteSet is a parameter. In our preliminary experiments, we tried several values for this parameter, which demonstrated the superiority of working with more than one solution.

At each iteration of BuildActList (see Algorithm 2), an eligible activity is selected, until a complete activity list is obtained. An activity is called 'eligible' when all its predecessors have been selected. Greedy activity selection would involve using the best solution found so far as the *reference* for this selection – that is, selecting the first eligible activity in that list. In order to randomize the selection, we will randomly choose among the elite set the solution that will serve as the reference; this selection is performed by the function SelectSolution. An elite solution remains the reference in the following nit ∈ [nitmin ; nitmax] iterations (randomly chosen). Additionally, to add even more randomness, we also include in SelectSolution the possibility that the eligible activity is either chosen according to its latest

Algorithm 3 SelectSolution(EliteSet)

```
Draw p ∈ [0; 1]
if p < pLFT then
  reference = "LFT"
else if p < pLFT + pRandom then
  reference = "random"
else
  A reference solution is randomly drawn from EliteSet
  if p < pLFT + pRandom + pInverse then
    reference = inv(reference)
  end if
end if
Return reference solution
```

finish time or that it is chosen randomly; these latter two options are only applied in a small fraction pLFT and pRandom, respectively; in this case nit = 0 (other values were examined but led to worse results). In the first iterations of our GRASP, when the elite set is not full, we set the values of pLFT and pRandom to 95% and 5%, respectively.

In order to introduce diversity into the procedure, we have included the possibility of yet another reference solution in the function SelectSolution (see Algorithm 3), namely the *inverse* of a list in EliteSet, with probability pInverse, via function inv. In this case, a list L from EliteSet is chosen, and the next activity in BuildActList is an eligible activity with highest (rather than lowest) position in L.

We have implemented a local-search procedure (in function LocalSearch) based on the concept of *justification*, which is a stepwise procedure for reducing the length of a deterministic schedule (using the deterministic durations in $\mathbf{d}^*$); more details can be found in Valls et al. (2005). This implementation is referred to as 'LS1'. We also investigate the application of a two-point crossover for permutations according to an implementation by Hartmann (1998). The input lists are the lists corresponding with the two schedules before and after justification; the resulting method is called 'LS2'. When LS2 is used, lines 5 and 6 in Algorithm 1 are altered, as the output of LocalSearch already contains a string.

5. Computational results for the expected makespan

In this section, we compare different versions of our GRASP-implementation with expected-makespan objective in order to evaluate the quality of the overall algorithm and its individual

elements (Section 5.1), and we provide a comparison of our algorithmic performance with other recently proposed algorithms (Section 5.2).

5.1 Details of our GRASP-implementation

We measure the quality of heuristic algorithms for solving the SRCPSP by the percentage distance of $E[s_n(\mathbf{D};\Pi(L))]$ (approximated using 1,000 replications, independent from the ones used in the optimization phase) from the critical-path length of the project with deterministic mean durations d_i^*. These percentages are averaged over all the instances of the set j120. In the literature on heuristics for the deterministic RCPSP, it is common (see, e.g., Hartmann and Kolisch, 2000) to impose a limit on the number of generated schedules, to facilitate comparison of different algorithms regardless of the computer infrastructure. We use two limits on the number of schedules: 5,000 and 25,000. Due to the particularities of activity-based policies (an activity cannot be scheduled before a previously scheduled activity), it turns out (see Ballestín, 2007) that activity-based policies are about twice as fast as the deterministic serial SGS. Consequently, we count one scheduling pass of an activity-based policy as 0.5.

The first line of Table 1 shows the results of the final version of our algorithm, simply called 'GRASP', where pInverse = 0, pRandom = 0.05 and LS2 is used. The second line, labelled 'Basic', pertains to an implementation without local search and without descriptive sampling. 'Basic+DS' refers to the inclusion of descriptive sampling (DS) in Basic. 'GRASP-LS1' is GRASP with LS1 instead of LS2. In 'Inverse', we set pInverse = 0.05 and pRandom = 0 to study whether the use of inverse solutions from EliteSet to attain diversity

Distribution	U1		U2		Exp		B1		B2	
# schedules	5,000	25,000	5,000	25,000	5,000	25,000	5,000	25,000	5,000	25,000
GRASP	46.84	45.21	72.58	70.95	114.42	112.37	47.17	45.60	75.97	74.17
Basic	50.57	48.68	76.55	74.78	118.72	117.20	50.70	48.99	79.49	77.64
Basic+DS	50.12	48.33	75.76	73.83	117.36	115.34	50.49	48.61	78.96	77.04
GRASP-LS1	49.17	47.73	76.68	74.93	121.07	119.02	49.75	48.14	80.50	78.62
Inverse	46.93	45.35	72.67	70.97	114.40	112.46	47.25	45.58	76.00	74.22
100 repl	49.81	46.73	75.36	71.76	116.76	112.36	50.20	47.18	78.63	75.00
500 repl	53.04	49.53	79.59	75.05	122.69	116.16	53.40	49.89	83.06	78.26
1,000 repl	53.79	51.15	80.50	76.95	123.70	118.77	54.15	51.51	83.97	80.28

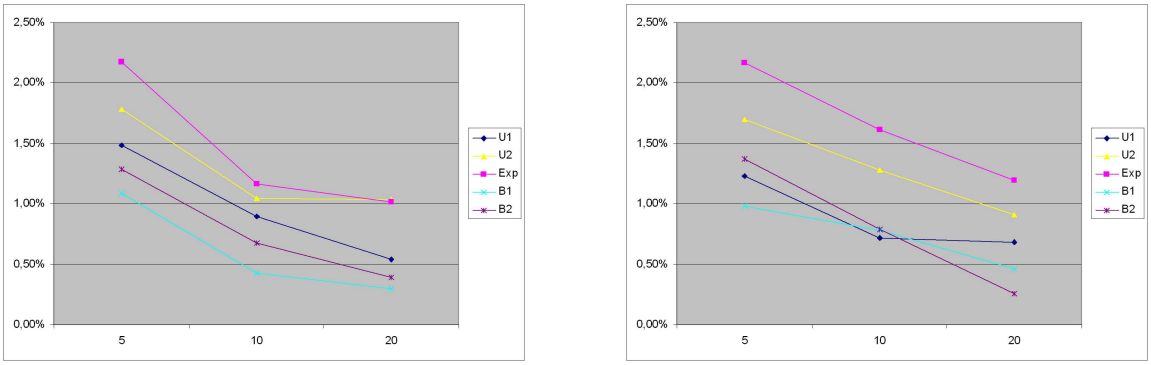
Table 1: Percentage distance of $E[s_n(\mathbf{D};\Pi(L))]$ (approximated using 1,000 replications) from the critical-path length of the project with deterministic mean durations d_i^*.

is useful. Finally, the last three lines of the table give the computational performance of GRASP with 100, 500 and 1,000 replications per examined policy rather than just 10.

We observe that the largest improvement is achieved by changing from LS1 to LS2. The straightforward use of justification improves the quality of Basic+DS only when activity-duration variability is low (U1-B1) but worsens it in the other cases. However, LS2 manages to improve the results in all cases. Inversely using elite solutions to introduce diversity is not beneficial for this dataset. Nevertheless, we would argue that it is a good way to diversify the search and we intend to explore other ways to implement this intuition in the future.

Adopting a low number of replications (10 rather than 100 or more) for evaluating each examined policy during the search procedure (line 7 in Algorithm 1) turns out to yield considerably better results (as was hinted at at the end of Section 3). For a low number, the search procedure is able to generate and evaluate more policies in the same time, or in our case, within the same schedule limit. Consequently, when computational effort matters, the best number of replications for evaluation during the search turns out to be rather low, leading to low accuracy but a higher number of scanned solutions, which usually results in a better final outcome. For evaluating the quality of the final policy that is output by the algorithm (the result of line 12 of Algorithm 1), on the other hand, accuracy is of course the most important characteristic, and so we use 1,000 replications.

Finally, the inclusion of the DS also (slightly) improves the results of the basic algorithm. The improvement in the average deviation from the critical-path length that is obtained by the incorporation of DS (compared to random sampling) was computed for 5, 10 and 20 replications; the results for a schedule limit of 5,000 and 25,000 schedules can be found in Figure 3. The gain obtained by DS tends to decrease as the number of replications increases, both for 5,000 and 25,000 schedules, and this trend is independent of the duration distribution. Specifically, the gain amounts to 2% with five replications for some distributions, making the incorporation of DS clearly worthwhile. Based on the data presented so far, we can recommend the use of descriptive rather than simple random sampling in heuristic search especially when the number of replications is low, in which case the potential improvement appears to be significant. Perhaps more important than the DS technique itself is the fact that this confirms the value of research regarding the implementation of the sampling in heuristic algorithms with replications, an observation that was also made by Gutjahr et al. (2000), and that is in line with earlier literature on the simulation of activity networks (Avramidis et al., 1991; Grant, 1983; Sullivan et al., 1982).



(a) 5,000 schedules.

(b) 25,000 schedules.

Figure 3: Computational results for descriptive sampling.

In order to further justify the algorithmic design choices for our GRASP-implementation, we include Table 2, where we summarize our findings after re-running the experiments that led to Table 1 on a totally different dataset. We used RanGen1[2] to generate a dataset with ten scheduling instances with $n = 91$ and $K = 3$ for each of the parameter settings: order strength $OS = 0.25, 0.50, 0.75$; resource factor $RF = 0.45, 0.9$; and resource constrainedness $RC = 0.3, 0.6$ (for more details about these parameters, we refer to Demeulemeester et al., 2003). We find that the attainable solutions have makespans that are more distant from the critical-path length, leading to much higher deviations than the ones in Table 1. This is not uncommon, a similar behavior can be observed in Debels et al. (2006) for the RCPSP, where instances from RanGen1 with 75 activities lead to a deviation of about 270%, whereas for j120 a deviation of between 30% and 35% was obtained. These results support the robustness

[2]Available at http://www.projectmanagement.ugent.be/rangen.php.

Distribution	U1		U2		Exp		B1		B2	
# schedules	5,000	25,000	5,000	25,000	5,000	25,000	5,000	25,000	5,000	25,000
GRASP	312.41	307.68	347.81	341.85	389.31	384.48	312.30	307.82	348.86	343.82
Basic	321.78	316.09	356.33	350.63	396.98	390.39	321.10	315.32	355.13	349.80
Basic+DS	319.78	314.15	353.43	348.30	394.82	388.63	320.43	314.75	354.95	349.29
GRASP-LS1	314.56	310.65	356.40	351.88	407.44	401.44	315.79	311.06	359.64	354.77
Inverse	311.59	307.61	347.41	342.67	389.97	384.66	312.65	307.87	349.39	344.68
100 repl	319.47	311.88	354.87	346.25	395.05	386.78	319.85	312.69	355.67	347.50
500 repl	328.75	321.57	364.11	357.45	405.50	398.76	329.02	322.14	365.45	358.57
1,000 repl	330.63	326.57	365.90	361.87	406.96	402.96	330.89	326.85	367.31	363.30

Table 2: Percentage distance of $E[s_n(\mathbf{D}; \Pi(L))]$ from the critical-path length of the project with deterministic mean durations for the RanGen-dataset.

of the algorithm, since the relative ranking of the different lines in the two tables is the same. The only difference is that inclusion of the inverse algorithm leads to slightly better results for U1, which confirms our suspicion that diversification can be worthwhile in some cases.

5.2 Comparison with state-of-the-art algorithms

We are now ready to compare our GRASP-algorithm with other SRCPSP-algorithms from the literature. First of all, we consider the genetic algorithm (GA) of Ballestín (2007), which uses the same dataset and schedule limit, with distributions U1, U2 and Exp. Table 3 provides a comparison; we can see that GRASP outperforms the GA in all cases. If we return to Tables 1 and 2, even our Basic algorithm does better than the GA, which highlights the improvement obtained by adding the descriptive sampling and LS2.

Secondly, we consider the tabu search (TS) and simulated annealing (SA) procedures developed by Tsai and Gemmill (1998), who evaluate algorithmic performance on the Patterson dataset (Patterson, 1984), with adaptations for obtaining stochastic (Beta) activity durations. As a measure of the quality of their algorithms, the authors report the deviation from an approximate lower bound. Table 4 shows their results, obtained on a personal computer with 166 MHz; SA2 and TS2 differ from SA1 and TS1 only in the parameters settings; the two final columns contain the results of our GRASP-algorithm on the same problem set. The GRASP-algorithm with a limit of 5000 schedules outperforms both the SA and the TS in quality and in time, even if we take into account the difference in computer infrastructure. The small difference between 5,000 and 25,000 schedules might be due to the fact that the solutions found are near-optimal.

Golenko-Ginzburg and Gonik (1997) test their algorithms on only one instance, which has 36 activities and a single resource type; Table 5 contains their and our results for this instance for three duration distributions. The authors do not report running times for the procedures but only point out that the algorithm that uses an exact procedure to solve consecutive multi-dimensional knapsack problems (Heuristic 1) needs much more time than the algorithm

Distribution	U1		U2		Exp	
# schedules	5,000	25,000	5,000	25,000	5,000	25,000
GA	51.94%	49.63%	78.65%	75.38%	120.22%	116.83%
GRASP	46.84%	45.21%	72.58%	70.95%	114.42%	112.37%

Table 3: Comparison between GRASP and GA.

Algorithm	SA1	SA2	TS1	TS2	GRASP (5,000)	GRASP (25,000)
Above approximate lower bound	3.40%	2.27%	3.71%	2.54%	2.01%	1.96%
Average time (s)	10.804	21.414	5.834	11.290	0.92	4.24

Table 4: Comparison between GRASP (with 5,000 and 25,000 schedules) and the algorithms of Tsai and Gemmil (1998).

Distribution	Beta	Uniform	Normal
Heuristic 1	433.88	448.49	448.85
Heuristic 2	447.98	461.35	461.58
GRASP (5,000)	408.75	427.64	422.04
GRASP (25,000)	403.16	424.28	415.40

Table 5: Expected makespan for the instance from Golenko-Ginzburg and Gonik (1997).

that solves these problems heuristically (Heuristic 2). Obviously, no strong conclusions can be drawn based on only one instance, but the difference between the algorithms is quite large, especially with Heuristic 2. Stork (2001) also tests his exact algorithm (branch-and-bound) on the same instance, but only for the Uniform distribution. He obtains an expected makespan of (rounded) 434 when the branch-and-bound is truncated.

6. Correlations and trade-offs

In this section, we investigate how the different statistics (expected makespan, probability of meeting a due date, etc.) behave relative to one another. In order to approximate the correlations, we need a convenient sample of solutions. It is possible to generate a purely random sample, but these frequently turn out to have a very large makespan, and obviously also a bad service level and tardiness. Therefore, we use Ballestín's (2007) GA to generate a sample of solutions with acceptable-to-good makespan values, which ensures solutions for most 'interesting' levels of the makespan (from 'not good but not very bad either' to 'excellent'). More specifically, we take the first 1,500 solutions generated by the GA itself (after crossover and mutation), the solutions in the initial population are not included. We do not use the GRASP-algorithm that is the subject of the first sections of this paper since this algorithm is used for all optimization runs, and we prefer to obtain all other information regarding the instances, such as due dates, correlations, etc., by other means than via the algorithm of which the performance is tested.

6.1 Expected makespan versus service level

The first relationship to be examined is that between service level and expected makespan, the first being a probability, the second a measure of schedule length. It is tempting to simply investigate the correlation between $E[s_n]$ and $Pr[s_n \leq \delta]$, for given values of δ. This approach has some disadvantages, however. First of all, it is difficult to choose appropriate due dates δ for an entire dataset: it may be more appropriate to have (a) different value(s) per instance. Second, since the service level is expressed as a percentage, it is not the most convenient quantity to rely on when computing correlations, because quite a number of solutions may have 0% or 100% service level. Therefore, we calculate for each instance the due date δ associated with given service levels and investigate the correlation between δ and $E[s_n]$ (so δ is an estimate of a quantile, based on the order statistics of the makespan sample generated via the GA). Table 6 contains the determination coefficient, which characterizes the linear relation between the two quantities (note that this coefficient is the square of the correlation coefficient).

We observe that the larger the variance, the smaller the correlation: the distribution with the smallest correlation is clearly Exp, followed by U2 and B2. The correlation is smallest in the tail and larger in the 'middle' of the domain. We conclude that, for most practical purposes, it is *not* necessary to work with service levels: optimal (or good) expected-makespan solutions *automatically* perform very well on the service-level objective. Only perhaps in some extreme cases will it be interesting to optimize the service level instead of the expected makespan, namely for very high levels ($\geq 99\%$) and large duration variability.

probability	U1	U2	Exp	B1	B2
50%	99.91%	99.60%	97.66%	99.89%	99.45%
	0.11	0.37	1.67	0.12	0.45
75%	99.84%	99.35%	96.94%	99.83%	99.22%
	0.18	0.60	1.96	0.19	0.58
90%	99.62%	98.41%	93.81%	99.61%	98.21%
	0.42	1.74	4.38	0.35	1.33
99%	98.21%	94.07%	73.91%	97.89%	92.62%
	1.96	5.11	13.72	1.92	5.00

Table 6: Determination coefficients for the relationship between due date and expected makespan, for four service levels (in the first column): average (first value) and standard deviation (second value) over the dataset. The columns correspond with the five duration distributions.

One should also take into account that the smaller correlations in these cases may be due in part to the difficulty in estimating the service levels (simulation of rare events) (cf. Section 5, especially Figure 2).

As a result of the foregoing conclusion, service-level bounds can be ignored as a representation of risk averseness: assuming near-perfect correlation, either the (unconstrained) solution with minimum expected makespan respects the service-level bound, or no solution offers a service level above the threshold. One additional remark is in order here: as noted in Section 3, for comparable precision one needs a considerably higher number of replications for the service level than for the expected makespan, so, other things being equal, the same number of replications can be expected to favor the selection of better solutions in the case of the makespan objective.

We are now ready to investigate the trade-off between due date and service level. The results are displayed in Figure 4; in line with the previous paragraphs, optimal (or at least high-quality) service levels are set via expected-makespan optimization. We observe close similarities between the distributions with similar variance (U1-B1 and U2-B2). The graphs also 'flatten out' as variability increases. Specifically, the service level changes drastically when the deadline changes for U1 and B1. For these low-variability distributions, a clear 'S'-shape is discerned, which shows that both for very high and very low service levels, the necessary improvement in average makespan to obtain a given service-level improvement

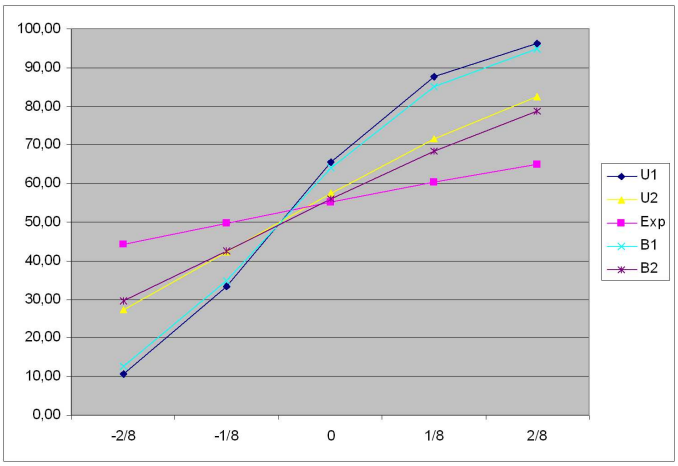


Figure 4: The trade-off between due date (abscissa) and service level (ordinate). Five due dates are considered, namely $\mathsf{min} - (2/8)(\mathsf{max} - \mathsf{min})$, $\mathsf{min} - (1/8)(\mathsf{max} - \mathsf{min})$, min, $\mathsf{min}+(1/8)(\mathsf{max}-\mathsf{min})$ and $\mathsf{min}+(2/8)(\mathsf{max}-\mathsf{min})$, where '$\mathsf{min}$' and '$\mathsf{max}$' are the lowest and highest makespan realization of a good GA-solution (for a high number of replications).

is higher than in the 'bulk' of the makespan spread, presumably because less solutions correspond to very high and low makespans. The trend is almost linear for U2, B2 and Exp, with the shallowest slope for Exp. These figures are averages over 600 instances, so the behavior may be different for individual instances.

6.2 Expected makespan versus variance

We first include a discussion on delivery dates (Section 6.2.1) and then present computational results (Section 6.2.2).

6.2.1 Delivery dates

For the benefit of risk averseness, one of the options envisaged is to impose lower bounds on the makespan variance. In principle, we can simply eliminate a solution if it does not respect the constraint. A possible problem is that, since our search procedure looks for objective-function improvements, it may generate *only* non-permissible solutions. We therefore proceed as follows: the *altered* makespan s_n^{alt} corresponding with an activity list L is obtained as

$$s_n^{alt}(\mathbf{D}; \Pi(L)) = \max\{s_n(\mathbf{D}; \Pi(L)); \Delta\}, \tag{1}$$

where Δ is an artificial *delivery date* for the schedule; higher Δ leads to higher expected makespan but lower variance. For a given upper bound on the variance, we find the lowest value of Δ such that the bound is respected (via binary search). Since we evaluate the performance measures by means of sampling, the computations corresponding with Equation (1) are straightforward (the max-operator is applied to known numbers for each sample). As an example, for four makespan realizations $s_n = 100$, 101, 103 and 104, Table 7 contains the quantities $s_n^{alt}(\mathbf{d}, \Pi(L))$. One activity list leads to multiple pairs $(E[s_n^{alt}], \text{var}[s_n^{alt}])$, dependent on Δ. For a given threshold (upper bound) on the variance, however, only one of those pairs comes out best, namely the pair with lowest $E[s_n^{alt}]$ such that $\text{var}[s_n^{alt}]$ does not exceed the threshold.

6.2.2 Computational results for the relationship variance/expected makespan

In this section, we investigate the relationship between the expected makespan and the makespan variance. For the same dataset as in Section 6.1, we obtain results quite different than before: average coefficients of determination are between 62% and 69% (see Table 8). Interestingly, the highest values occur for the Exponential distribution, while these were

	$\Delta =$	99	100	101	106
$s_n = 100$		100	100	101	106
101		101	101	101	106
103		103	103	103	106
104		104	104	104	106

Table 7: Altered makespan s_n^{alt} corresponding with four different values for the delivery date Δ. Each column contains one sample of altered makespans.

lowest for the service level (see Table 6). In part, the latter phenomenon may be due to the fact that the expectation of an Exponential variable is equal to its standard deviation, so that for a given longest path of activities in the schedule, a higher expectation will also imply a higher variance (although the same is true, but to a lesser extent, for the other distributions; in the context of resource-constrained scheduling, one should also generally avoid overly focusing on individual paths in the precedence network).

It can be concluded that these correlations are insufficiently high to neglect the variance, and we anticipate an actual expectation/variance trade-off. This trade-off is examined in Figure 5, where the expected makespan obtained by the GRASP-algorithm is plotted as a function of an upper bound imposed on the variance (actually on the standard deviation). The graphs are very similar in shape for all five distributions, but there are differences, mainly in the 'jumps' in the expected makespan corresponding with a step from one k-value to the next. For example, the jump for U1 is five units of expected makespan from $k = \frac{1}{4}$ to $k = 1$, while it is 45 for Exp. Consequently, the decision maker needs to 'sacrifice' a considerable increase in makespan expectation if he/she wants to restrict the makespan variance in cases where activity duration variance is high. However, this loss is negligible when the variances of the D_i are low, unless the restriction is severe (corresponding to low k-values). This observation matches intuitive expectations, but has now been demonstrated numerically as well.

U1	U2	Exp	B1	B2
62.12%	63.43%	68.67%	64.19%	65.50%
17.36	17.17	14.08	15.94	16.60

Table 8: Determination coefficients for the relationship between standard deviation and expected makespan: average (first value) and standard deviation (second value) over the dataset. The columns correspond to the five duration distributions.

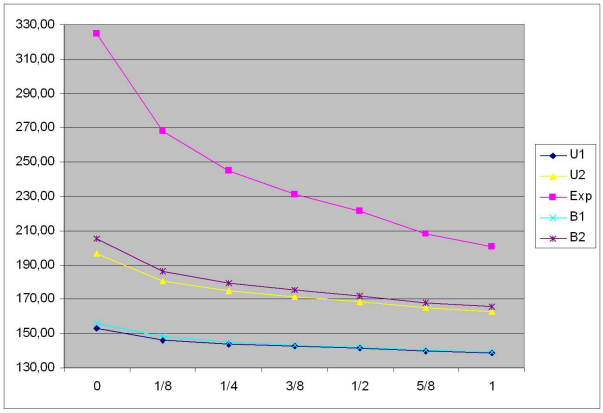


Figure 5: The trade-off of variance (abscissa) versus expected makespan (ordinate). Let 'min' represent the minimum standard deviation over all solutions that were examined by the GA in a search for the minimum expected makespan (unconstrained). The upper bounds imposed on the standard deviation are $k \times \mathsf{min}$, with $k = 0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}$ and 1.

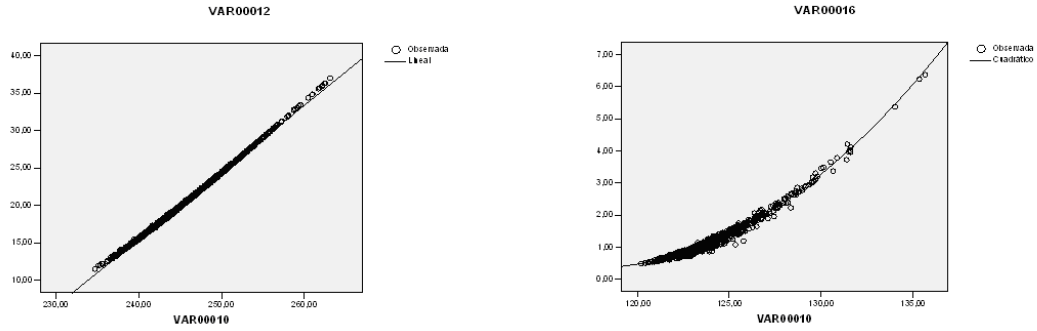
6.3 Expected makespan versus expected tardiness

We investigate the minimization of the expected tardiness $E[\max\{0; s_n - \delta\}]$ once the manager has fixed a deadline δ. We are obviously mainly interested in δ-values such that

$$\min_{\mathbf{d}}\{s_n(\mathbf{d}; \Pi)\} < \delta < \max_{\mathbf{d}}\{s_n(\mathbf{d}; \Pi)\}, \tag{2}$$

where optimization in the first and third term is performed over all possible duration-realization vectors $\mathbf{d}$ in the sample used, and Π is any activity-based policy (otherwise, we either have an optimal objective of zero or we simply minimize expected makespan). In order to examine these cases, we compute 'minmin' as the minimum of the minimization term in Equation (2), taken over all policies Π examined by the GA, and similarly 'minmax' as the minimum of the maximization term in (2). We wish to investigate especially $\delta \in]\mathsf{minmin}; \mathsf{minmax}[$; other values often turn out to admit policies with *all* makespan realizations either higher or lower than the deadline. More specifically, we select three values for δ, namely $\delta_1 = \mathsf{minmin} + (\mathsf{minmax} - \mathsf{minmin})/2$, $\delta_2 = \mathsf{minmin} + 5(\mathsf{minmax} - \mathsf{minmin})/8$ and $\delta_3 = \mathsf{minmin} + 3(\mathsf{minmax} - \mathsf{minmin})/4$.

We find that, dependent on the value of the deadline, two types of relation between the expected makespan and the expected tardiness can be encountered, either a linear or a quadratic one; this is illustrated in Figure 6. A high determination coefficient in a linear or in a quadratic relationship between $E[s_n]$ and $E[\max\{0; s_n - \delta\}]$ would imply that, for all practical purposes, optimization of the first quantity will suffice to optimize the second



(a) A linear example.

(b) A non-linear example.

Figure 6: Expected makespan (abscissa) versus expected tardiness (ordinate).

as well. It turns out that on fitting a quadratic equation to the data (which obviously includes a linear relationship), the determination coefficients are above 95% in all cases, and the majority even exceed 99%; for the sake of brevity, we omit the table displaying all coefficients.

The literature has already proposed algorithms for minimizing the (expected) makespan, but not for optimizing the other objective functions, which are nevertheless also important. The key idea of Section 6 is that it is not so important to develop specific algorithms for these functions, because the existing algorithms for the makespan tend to optimize the other objectives at the same time. While our observations may not amount to irrefutable proof, they nevertheless provide considerable evidence in favor of this thesis: the levels of correlation achieved indicate that by searching for a scheduling policy with lowest expected makespan, one usually simultaneously minimizes the expected tardiness and maximizes the service level. Obviously, dedicated algorithms could perhaps obtain the same results in less time or achieve higher-quality outcomes with the same computational effort.

7. The distribution of the makespan realizations of a given policy

In a deterministic setting, the decision maker knows exactly when the project will be finished and can make decisions based on this information. In a stochastic environment, he/she can only rely on the expected makespan, which is a very limited piece of information given that many different makespan realizations can actually occur. Clearly, knowledge of the entire distribution of possible makespan realizations $G(t;\Pi(L)) : \mathbb{R} \mapsto [0;1] : G(t;\Pi(L)) =$

$Pr[s_n(\mathbf{D}; \Pi(L)) \leq t]$ is much more informative. Our goal in this section is to better understand the shape of this distribution. Our first step is to calculate two descriptive measures for the shape and symmetry of a policy's makespan distribution, its skewness and its kurtosis. Approximations for these values are collected in Table 9. The first (second) line shows the average of the (absolute) skewness of the different instances. The third line represents the fraction of the instances with positive skewness. The remaining lines display the same results for the kurtosis.

The data drawn from both Uniform distributions are very symmetric, and even the number of instances with positive and negative skewness is around 50%. The absolute value of the kurtosis is small, although there are more instances in which the kurtosis is negative. At any rate, if we pay attention to the absolute values, the data could stem from a Normal distribution (which has zero skewness and kurtosis). Interestingly, the increase in variability from U1 to U2 hardly affects the measures. The situation is slightly different for the Beta distributions: the B2 figures are very similar to those of U1 and U2, but for B1 we observe data that are less symmetric (long right-sided tail), yet more unbiased in the case of positive and negative skewness. We can still consider the Normal distribution as a possible model for these data. Finally, the Exponential distribution data have a long right-sided tail (positive skewness) and a prominent peak (positive kurtosis). The Normal distribution should not be able to capture these data.

The next step in our attempt to characterize the distribution of the makespan realizations is to find a known distribution that adequately fits the obtained data, for which many choices are available. Based on the previous measures and histograms of the data (e.g. Figure 7 for instance j1201_1 if the durations stem from U1), we tried to fit a Normal distribution $\mathcal{N}(\mu, \sigma^2)$ to the data, with μ the average of the makespan realizations and σ^2 the (sample) variance. One method of determining whether a certain distribution appropriately fits the data is a

Distribution	U1	U2	Exp	B1	B2
skewness	0.006	−0.009	0.492	0.143	0.014
|skewness|	0.067	0.069	0.492	0.148	0.074
% inst.> 0	51.33%	43.67%	100.00%	92.00%	52.67%
kurtosis	−0.030	−0.037	0.440	0.023	−0.051
|kurtosis|	0.128	0.122	0.450	0.130	0.133
% inst.> 0	36.50%	35.67%	92.67%	49.33%	33.00%

Table 9: Skewness and kurtosis for the different distributions.

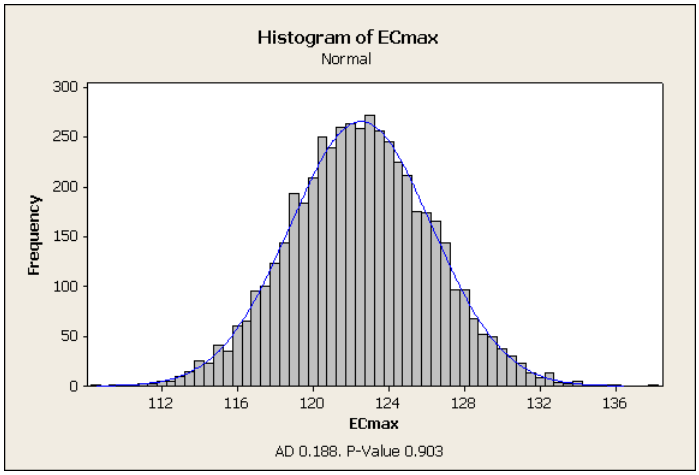


Figure 7: Histogram of the different makespan realizations of a solution for j1201_1.

hypothesis contrast or test:

$$H_0 : G = \mathcal{N}(\mu, \sigma^2) \qquad \text{(null hypothesis)}$$
$$H_1 : G \neq N(\mu, \sigma^2) \qquad \text{(alternative hypothesis)}$$

The hypothesis can be contrasted by calculating a statistic of a sample of G. We will use the Anderson-Darling (A-D) statistic A^2 (D'Agostino, 1986; Linnet, 1988):

$$A^2 = -M - \frac{1}{M}\sum_{i=1}^{M}(2i-1)(\ln(G(y_i) - \ln(G(y_{M+1-i})))),$$

where $\{y_i\}_{i=1}^{M}$ are M (not necessarily different) makespan realizations obtained from M replications. This test can be applied to any distribution, although critical values of the A-D statistic under the null hypothesis have only been tabulated for a limited number of distributions. Table 10 shows the fraction of instances (out of the 600) for which the A-D value is smaller than the 5% critical value, together with the average and the maximum of the statistic, and the percentage of instances in which it is larger than 1. According to the table, the Normal distribution can be used to model the distribution of the makespan realizations of the solution given by the GRASP, at least in most instances of U1, U2 and B2, and also in many instances (but quite fewer than in B2) of B1. We should add that the average of the A-D statistic in U1, U2 and B2 corresponds to a significance level larger than 0.2 and that many instances lead to a A-D statistic associated with levels over 0.5 (some are 0.8 and 0.9). The table also indicates that the Normal is of no use at all in the case of Exp.

Distribution	U1	U2	Exp	B1	B2
% A-D $<$ 5% crit. val.	92.67%	92.67%	1.00%	70.00%	90.67%
Average A-D stat.	0.404	0.413	3.032	0.648	0.433
Maximum A-D stat.	1.41	1.82	7.868	3.065	2.409
% inst. A-D $>$ 1	2.33%	2.00%	96.00%	15.00%	3.33%

Table 10: A-D values and hypothesis tests for the different distributions.

Distribution	U1	U2	Exp	B1	B2
% A-D $<$ 5% crit. val.	94.00%	94.67%	95.67%	95.00%	94.50%
Average A-D stat.	0.324	0.327	0.316	0.322	0.327
Maximum A-D stat.	1.105	1.106	0.956	1.782	1.304
% inst. A-D $>$ 1	0.17%	0.33%	0.00%	0.50%	0.50%

Table 11: A-D values and hypothesis tests for the different distributions, for the transformed data.

When data do not pass a normality test, it is common practice to transform them and apply the test to the transformed data. We have applied the Box-Cox transformation, given by

$$x = f_\lambda(y) = \begin{cases} (y^\lambda - 1)/\lambda, & \lambda \neq 0; \\ \ln y, & \lambda = 0. \end{cases}$$

The parameter λ is found via maximum likelihood. Table 11 contains the same information as Table 10 but with respect to the transformed data (note that the critical values are lower than in the original test, which is not always taken into account in statistical packages). The table shows that this transformation enables us to adequately fit most of the instances for all five the duration distributions: all averages of the A-D statistic are very similar and correspond to significance levels larger than 0.2. For most instances, a whole range of values for λ would lead to accepting the contrast; for the Uniform and Beta distributions, this range usually contains the value 1 ($\lambda = 1$ corresponds with no transformation); for Exp, the chosen value of λ is always in $[-1; 1]$ and mostly in $[-0.5; 0]$.

We conclude that it is generally possible to find known distributions that fit the makespan realizations of a given solution (a given policy) reasonably well, although the result may change if the parameters of the GRASP are altered (especially the number of replications or the limit on the number of schedules). Knowledge of an approximate distribution of the makespan realizations of an implemented policy enables the decision maker to compute values such as $Pr[a \leq s_n \leq b]$ for any a, b, and therefore constitutes a valuable piece of managerial information. In a study of the stochastic flow shop, Dodin (1996) uses Monte-Carlo sampling

to determine the makespan distribution of a given sequence. He argues that, for different duration distributions, the makespan of a given sequence becomes approximately Normal when a large number of jobs are involved (and positively skewed in case of large variance), but only based on a visual observation of the shape of the sample distributions, so without hypothesis testing.

In interpreting our results, a number of cautionary remarks are in order. Firstly, we have performed a large number of tests and therefore we can expect to find contrasts where the test fails even when H_0 is true: if we produce a large number of random samples of the same size from the same $\mathcal{N}(\mu, \sigma^2)$, around 5% of them would fail the A-D contrast with a critical value of 5%. Secondly, if a sample passes a test, this does not mean that H_0 is correct, only that the available information does not warrant rejecting it. However, in our tests other information supports our choice of distribution, e.g. the histograms and the small averages of the A-D statistic. In conclusion, it cannot be unequivocally stated that the data stem from (transformed) Normal distributions, but they do seem to provide good approximations.

Given this caveat, and with a reasonable degree of certainty, we conclude that the solutions obtained by the GRASP in the Exp-case do *not* follow a Normal distribution and that their distributions are therefore quite different from those in the case of e.g. U1, which can mostly be modeled by a Normal distribution. It should be pointed out that the makespan distributions are derived for specific choices for the duration distributions and under the assumption of independence between the durations. Since the parameters of the duration distributions have not been calibrated to real-world applications, it is not clear under which circumstances the statistical test results in Tables 10 and 11 remain valid for applications.

8. Summary

This article has investigated the incorporation of explicit recognition of variability into project planning by developing activity-based scheduling policies for the stochastic RCPSP. We have examined multiple possible objective functions for project scheduling with stochastic activity durations, and we have looked into the close connection between these different objective functions by means of computational experiments. Our findings provide considerable evidence for the thesis that for most practical purposes, it suffices to focus on the minimization of the expected makespan. We have proposed a GRASP-heuristic that produces high-quality solutions, outperforming the currently available procedures. The variance-reduction

technique of descriptive sampling has been applied and its benefits assessed. Finally, we have also studied the distribution of the makespan realizations for a given scheduling policy.

Acknowledgments

This research was partially supported by the Ministerio de Educación y Ciencia (Spain) under contract DPI2007-63100, and by research project G.0578.07 of the Research Foundation – Flanders (FWO) (Belgium).

References

Adlakha, V.G., V.G. Kulkarni. 1989. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR* **27**(3) 272–296.

Aiex, R.M., M.G.C. Resende, C.C. Ribeiro. 2002. Probability distribution of solution time in GRASP: an experimental investigation. *Journal of Heuristics* **8**(3) 343–373.

Ang, A., J. Chen, Y. Xing. 2006. Downside risk. *The Review of Financial Studies* **19**(4) 1191–1239.

Avramidis, A.N., K.W. Bauer Jr., J.R. Wilson. 1991. Simulation of stochastic activity networks using path control variates. *Naval Research Logistics* **38**(2) 183–201.

Ballestín, F. 2007. When it is worthwile to work with the stochastic RCPSP? *Journal of Scheduling* **10**(3) 153–166.

Bayiz, M., C.J. Corbett. 2005. Coordination and incentive contracts in project management under asymmetric information. Working Paper CC31, Anderson Graduate School of Management, University of California, Los Angeles.

Blazewicz, J., J. Lenstra, A. Rinnooy-Kan. 1983. Scheduling subject to resource constraints – classification and complexity. *Discrete Applied Mathematics* **5**(1) 11–24.

Cho, J.G., B.J. Yum. 1997. An uncertainty importance measure of activities in PERT networks. *International Journal of Production Research* **35**(10) 2737–2757.

D'Agostino, R.B. 1986. Tests for the normal distribution. R.B. D'Agostino, M.A. Stephens, eds. *Goodness-of-Fit Techniques*. Marcel Dekker, 367–419.

Debels, D., B. De Reyck, B., R. Leus, M. Vanhoucke. 2006. A hybrid scatter search / electromagnetism meta-heuristic for project scheduling. *European Journal of Operational*

Research **169**(2) 638–653.

Demeulemeester, E., W. Herroelen. 2002. *Project Scheduling – A Research Handbook*. Kluwer Academic Publishers, Boston.

Demeulemeester, E., M. Vanhoucke, W. Herroelen. 2003. A random network generator for activity-on-the-node networks. *Journal of Scheduling* **6**(1) 13–34.

Dodin, B. 1996. Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops. *Computers & Operations Research* **23**(9) 829–843.

Elmaghraby, S.E. 1977. *Activity Networks: Project Planning and Control by Network Models*. Wiley.

Elmaghraby, S.E., Y. Fathi, M.R. Taner. 1999. On the sensitivity of project variability to activity mean duration. *International Journal of Production Economics* **62**(3) 219–232.

Feo, T.A., M.G.C. Resende. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization* **6**(2) 109–133.

Feo, T.A., M.G.C. Resende, S. Smith. 1994. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research* **42**(5) 860–878.

French, S. 1988. *Decision Theory. An Introduction to the Mathematics of Rationality*. Ellis Horwood Limited.

Gerchak, Y. 2000. On the allocation of uncertainty-reduction effort to minimize total variability. *IIE Transactions* **32**(5) 403–407.

Golenko-Ginzburg, D., A. Gonik. 1997. Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics* **48**(1) 29–37.

Graham, R.L. 1966. Bounds on multiprocessing timing anomalies. *Bell System Technical Journal* **45** 1563–1581.

Grant III, F.H. 1983. A note on "Efficiency of the antithetic variate method for simulating stochastic networks". *Management Science* **29**(3) 381–384.

Gutierrez, G.J., A. Paul. 2001. Robustness to variability in project networks. *IIE Transactions* **33**(8) 649–660.

Gutjahr, W.J., C. Strauss, E. Wagner. 2000. A stochastic branch-and-bound approach

to activity crashing in project management. *INFORMS Journal on Computing* **12**(2) 125–135.

Hagstrom, J.N. 1988. Computational complexity of PERT problems. *Networks* **18**(2) 139–147.

Hart, J.P., A.W. Shogan. 1987. Semi-greedy heuristics: an empirical study. *Operations Research Letters* **6**(3) 107–114.

Hartmann, S. 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics* **45**(7) 733–750.

Hartmann, S., R. Kolisch. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research* **127**(2) 394–407.

Herroelen, W. 2005. Project scheduling – theory and practice. *Production and Operations Management* **14**(4) 413–432.

Igelmund, G., F.J. Radermacher. 1983. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks* **13**(1) 1–28.

Jorion, P. 2000. *Value at Risk: The Benchmark for Controlling Market Risk*. McGraw-Hill.

Kerzner, H. 1998. *Project Management*. Wiley.

Kleywegt, A.J., A. Shapiro, T. Homem-De-Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12**(2) 479–502.

Kolisch, R., S. Hartmann. 1999. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. J. Weglarz, ed. *Project Scheduling – Recent Models, Algorithms and Applications*, Kluwer Academic Publishers, Boston, 147–178.

Kolisch, R., R. Padman. 2001. An integrated survey of deterministic project scheduling. *Omega* **29**(3) 249–272.

Kolisch, R., A. Sprecher. 1996. PSPLIB – a project scheduling problem library. *European Journal of Operational Research* **96**(1) 205–216.

Kouvelis, P., G. Yu. 1997. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers.

Leus, R., W. Herroelen. 2004. Stability and resource allocation in project planning. *IIE Transactions* **36**(7) 667–682.

Linnet, K. 1988. Testing normality of transformed data. *Applied Statistics* **37**(2) 180–186.

Ludwig, A., R.H. Möhring, F. Stork. 2001. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research* **102**(1–4) 49–64.

Möhring, R.H., F.J. Radermacher. 1989. The order-theoretic approach to scheduling: the stochastic case. R. Slowinski, J. Weglarz, eds. *Advances in Project Scheduling*, Elsevier, Chapter III.4.

Neumann, K., C. Schwindt, J. Zimmermann. 2002. *Project Scheduling with Time Windows and Scarce Resources*. Springer.

Newbold, R.C. 1998. *Project Management in the Fast Lane*. The St. Lucie Press/APICS Series on Constraints Management.

Patterson, J.H. 1984. A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. *Management Science* **30**(7) 854–867.

Portougal, V., D. Trietsch. 1998. Makespan-related criteria for comparing schedules in stochastic environments. *Journal of the Operational Research Society* **49**(11) 1188–1195.

Saliby, E. 1990. Descriptive sampling: a better approach to Monte Carlo simulation. *Journal of the Operational Research Society* **41**(12) 1133–1142.

Saliby, E. 1997. Descriptive sampling: an improvement over Latin hypercube sampling. S. Andradóttir, K.J. Healy, D.H. Withers, B.L. Nelson, eds. *Proceedings of the 1997 Winter Simulation Conference*, 230–233.

Schuyler, J. 2001. *Risk and Decision Analysis in Projects*. Project Management Institute.

Silver, E.A., D.F. Pyke, R. Peterson. 1998. *Inventory Management and Production Planning and Scheduling*. John Wiley & Sons.

Smith, P.G., D.G. Reinertsen. 1991. *Developing Projects in Half the Time*. Van Nostrand Reinhold.

Stork, F. 2001. Stochastic resource-constrained project scheduling. Ph.D. thesis, Technische Universität Berlin.

Sullivan, R.S., J.C. Hayya, R. Schaul. 1982. Efficiency of the antithetic variate method for

simulating stochastic networks. *Management Science* **28**(5) 563–572.

Tsai, Y.-W., D.D. Gemmill. 1998. Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research* **111**(1) 129–141.

Valls, V., F. Ballestín, S. Quintanilla. 2005. Justification and RCPSP: a technique that pays. *European Journal of Operational Research* **165**(2) 375–386.

Wollmer, R.D. 1985. Critical path planning under uncertainty. *Mathematical Programming Study* **25** 164–171.

Xie, G., J. Zhang, K.K. Lai. 2006. Risk avoidance in bidding for software projects based on life cycle management theory. *International Journal of Project Management* **24**(6) 516–521.